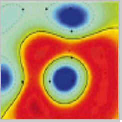


Protocol Inspection and State Machine Analysis (PRISMA)

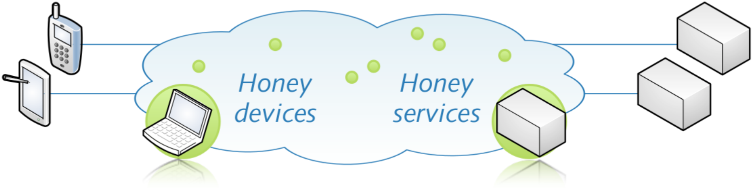
Tammo Krueger
Hugo Gascon
Konrad Rieck

Protocol Inspection and State Machine Analysis (PRISMA)

Tammo Krueger
Hugo Gascon
Konrad Rieck

19.6.2012 University of Göttingen

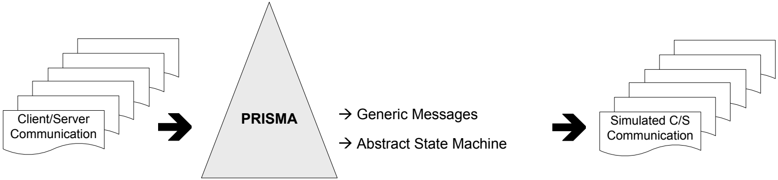
Proactive Security for Convergent Communication (PROSEC)



Proactive protection of services:

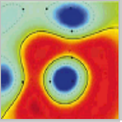
- Self-learning protocol analysis
- Deployment of “Honey-Services”
- Proactive protection of communication and attack detection

Motivation

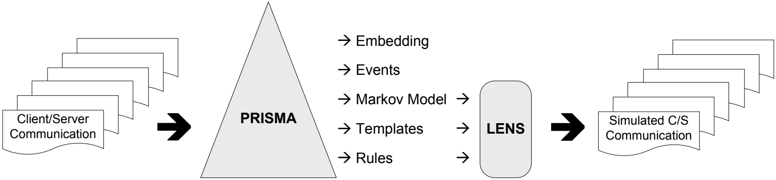


Given a pool of client/server communication infer generic messages and abstract state machine

1. To emulate services (honeypots)
2. Lure attackers
3. Gather information about threat potential



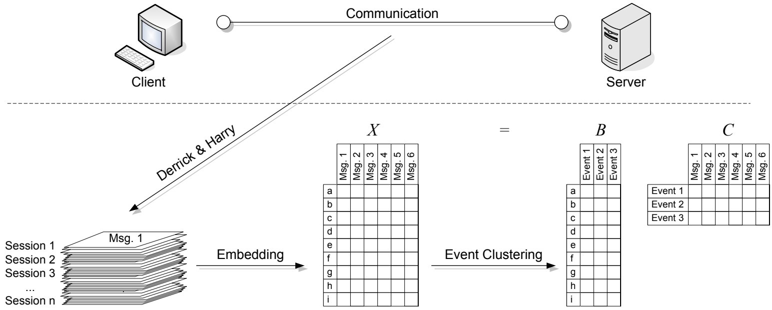
Motivation – Top-Down Model of Tasks

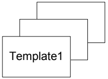


By embedding and event clustering approximate abstract state machine and message types:

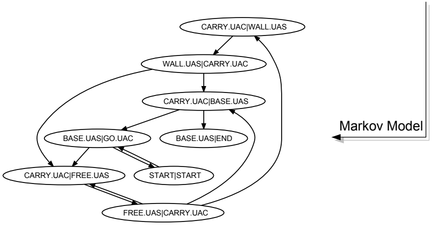
- Infer *Markov model* of the behavior
- Find inherent structure of the messages (*templates*)
- Gather information flow between states (*rules*)

System Overview

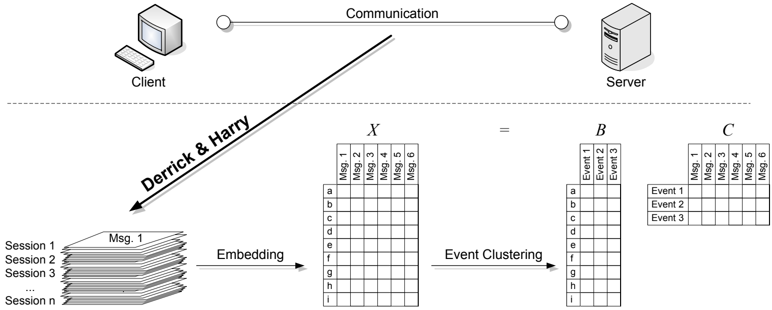


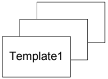



```
RULE srcField:0 dstField:0 type:ExactRule
RULE srcField:1 dstField:2 type:ExactRule
RULE srcField:3 dstField:1 type:ExactRule
RULE dstField:4 type:DataRule
data:678537408252460955, ...
RULE dstField:5 type:DataRule
data:252460957,252461242,252461608, ...
RULE dstField:6 type:DataRule
data:12004,12020,12018,12028,12002,12004
```

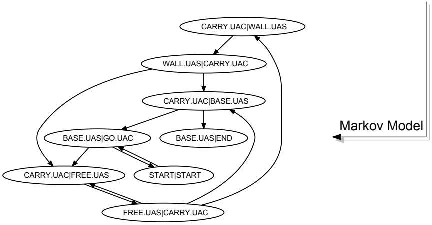


System Overview – Preprocessing






```
RULE srcField:0 dstField:0 type:ExactRule
RULE srcField:1 dstField:2 type:ExactRule
RULE srcField:3 dstField:1 type:ExactRule
RULE dstField:4 type:DataRule
data:678537408252460955, ...
RULE dstField:5 type:DataRule
data:252460957,252461242,252461608, ...
RULE dstField:6 type:DataRule
data:12004,12020,12018,12028,12002,12004
```



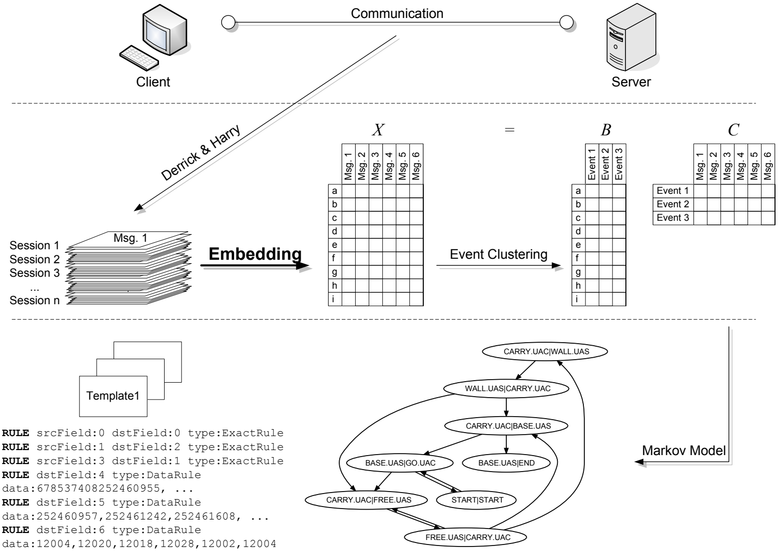
Preprocessing

- Data acquisition via `tcpdump`
- Tool chain needed, to process these binary dump files

  Derrick assembles packet contents based on the mature `libnids` library

  Harry concatenates packets to messages and extracts session information
- Data available for the next steps:
  1. messages as sequence of bytes
  2. sessions as sequence of messages
- Point 1 will be used in the *embedding* step
- Outcome of the *embedding* step will be used in the *clustering* step
- Point 2 and outcome of the *clustering* will be used in the *model building* step

System Overview – Embedding



Embedding

1. **N-grams:** Given the set of all possible n-grams over byte sequences $S = \{0, \dots, 255\}^n$, we define the embedding function $\phi : \{0, \dots, 255\}^\star \mapsto \mathbb{R}^{|S|}$ as

$$\phi(x) = (\phi_s(x))_{s \in S} \quad \text{with} \quad \phi_s(x) = \mathrm{occ}_s(x).$$

   Example ($n = 3$):

$$\phi(\text{"Hello"}) = (0, \dots, \overset{\text{Hel}}{1}, \overset{\text{ell}}{1}, \overset{\text{llo}}{1}, \dots, 0)^T \in \mathbb{R}^{16777216}$$

2. **Tokens:** Given a set of seperators *Sep* we can split the byte sequence into tokens; example ($Sep = \{␣\}$):

$$\phi(\text{"We'll meet again"}) = (0, \dots, \overset{\text{We'll}}{1}, \overset{\text{meet}}{1}, \overset{\text{again}}{1}, \dots, 0)^T \in \mathbb{R}^?$$

Dimension Reduction via Statistical Testing

- Embedding space high-dimensional but sparse
- Some dimension do not carry real information:
  - Fixed *protocol* tokens
  - Random, *volatile* tokens (cookies, nonces, ...)
- Focus the analysis by splitting the feature set F:

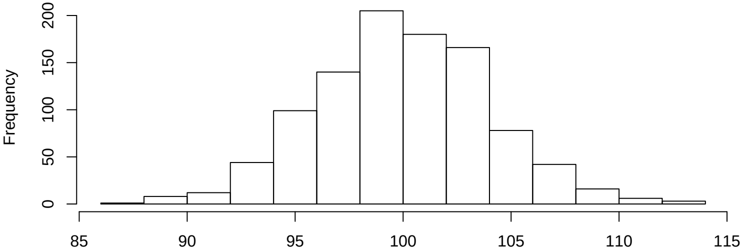
$$F = F_{protocol} \cup F_{alphabet} \cup F_{volatile}$$

- Keep features, which are not part of the protocol **and** are not volatile
- How to decide, whether a feature belongs to $F_{protocol}$ or $F_{volatile}$? Use **statistical testing!**

Anatomy of a Statistical Testing Procedure

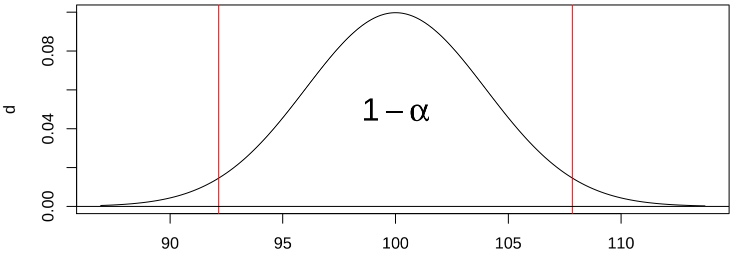
- Given measurements decide, whether we can accept or reject a hypotheses (H_0) in favor to an alternative (H_1) in a statistical sense
- Distribution assumption (parametric/non-parametric)
- Predefined significance level ($\alpha \in 0.01, 0.05, 0.1$)
- Test statistic
- p-value: probability to observe a value for the test statistic at least as extreme as the value that was actually observed given H_0 is true
- Decision rule: reject H_0 if p-value is smaller than the significance level

Statistical Test – Example I



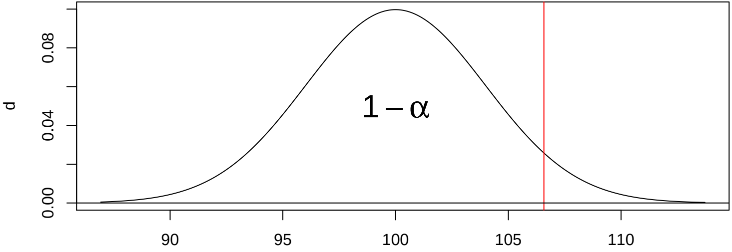
- Measure impact of fertilizer A on grain shaft size:
  1. collect N samples from several fields treated with fertilizer A
  2. record the mean size of these N samples per field
  3. plot the distribution and calculate mean μ_A and standard deviation σ_A
- Outcome: $\mu_A = 100cm, \sigma_A = 4cm$

Statistical Test – Example I



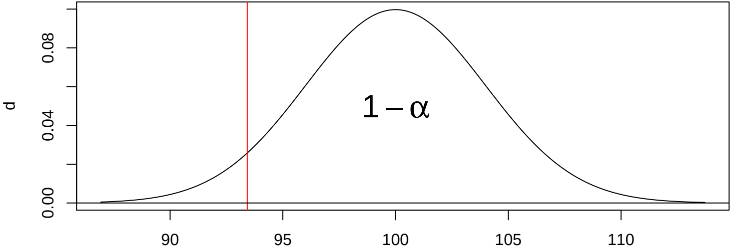
- Given a new measurement of a field x, can we determine with a given error level of α, whether it was treated with fertilizer A?
- Assume normal distribution $\rightarrow (1-\alpha) = 95\%$ of the data lies in the interval $[92.16, 107.83]$!

Statistical Test – Example I



- Different question: has new field x a bigger grain shaft size than the fields treated with fertilizer A?
- Assume normal distribution $\rightarrow (1-\alpha) = 95\%$ of the data lies in the interval $[-\infty, 106.57]$!

Statistical Test – Example I



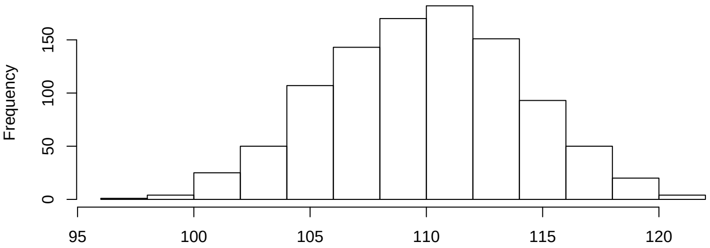
- Different question: has new field x a smaller grain shaft size than the fields treated with fertilizer A?
- Assume normal distribution → $(1-\alpha) = 95\%$ of the data lies in the interval $[93.42, +\infty]$!

Statistical Testing: What Can Go Wrong?

	H_0 is true	H_1 is true
Accept H_0	Right decision	Type II Error (β)
Reject H_0	Type I Error (α)	Right decision

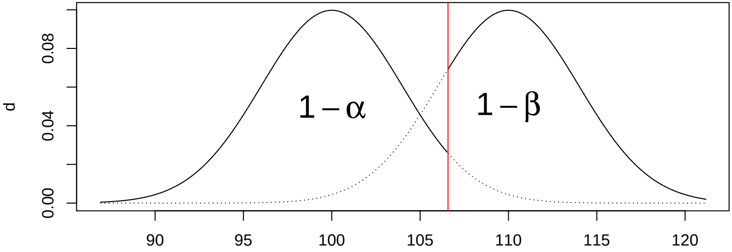
- Type I error controlled by *significance level* α
- Type II error is used to describe the *power* $(1 - \beta)$, i.e. the probability of correctly rejecting H_0

Statistical Test – Example II



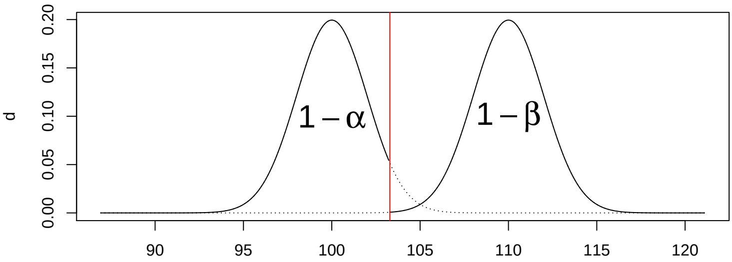
- Measure impact of new fertilizer B on grain shaft size:
  1. collect N samples from several fields treated with fertilizer B
  2. record the mean size of these N samples per field
  3. plot the distribution and calculate mean μ_B and standard deviation σ_B
- Outcome: $\mu_B = 110cm, \sigma_B = 4cm$

Statistical Test – Example II



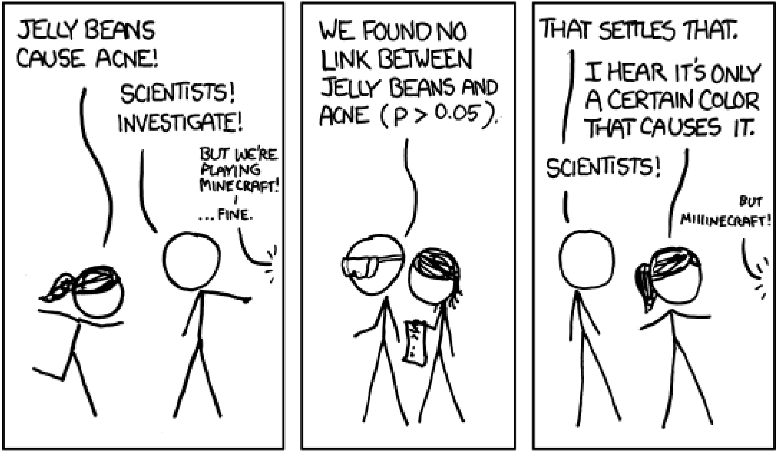
- Different question: has new field x been treated with fertilizer A or fertilizer B?
- Power $1 - \beta$ of test is directly connected to the significance level α!

Statistical Test – Example II



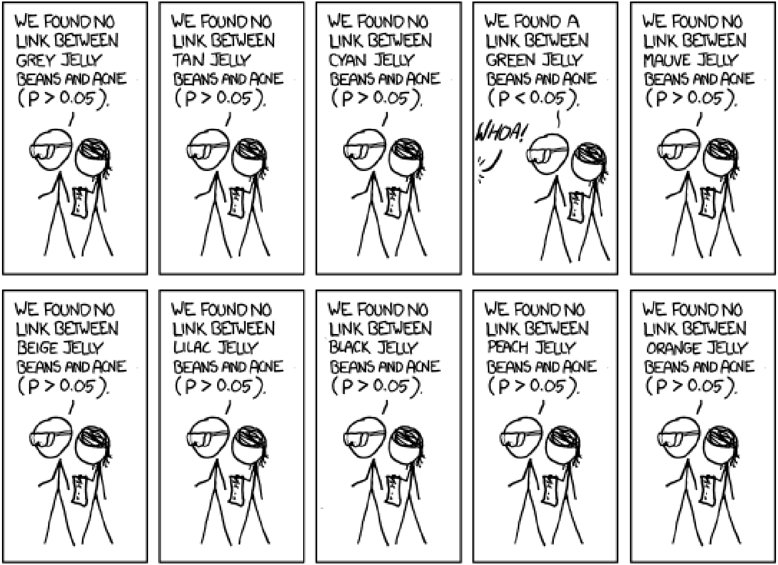
- How can we improve the power of the test?
- Increase the sample size N to lower the standard deviations σ_A and σ_B!

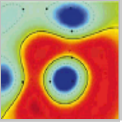
Statistical Testing: What Can Go Wrong? – Part 2



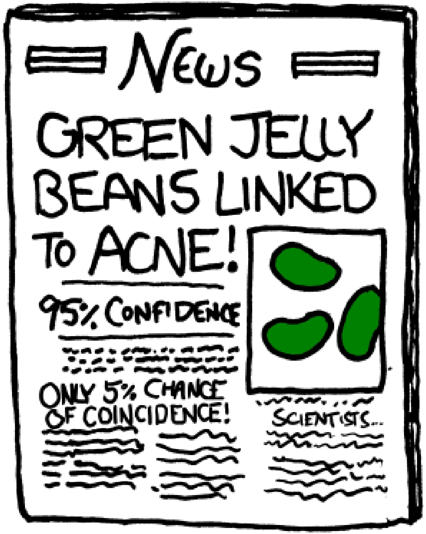
http://xkcd.com/882/

Statistical Testing: What Can Go Wrong? – Part 2

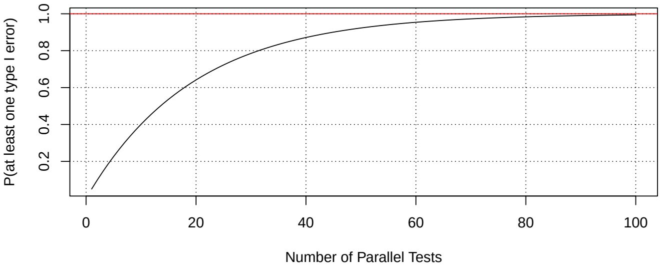




Statistical Testing: What Can Go Wrong? – Part 2



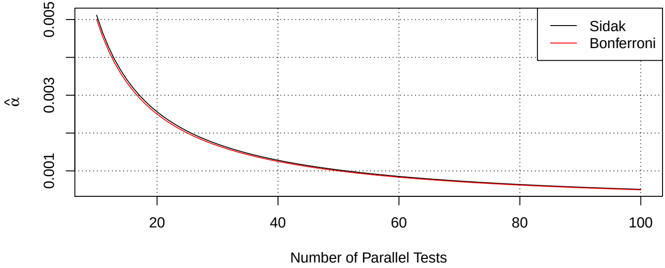
Multiple Testing



- In explorative data studies (e.g. micro-array experiments) a lot of tests are made in parallel
- For each of these k tests an error of type I can occur with probability α:

$$
\begin{aligned}
& P(\text{at least one type I error}) \\
= \quad & 1 - P(\text{no type I error}) \\
= \quad & 1 - (1-\alpha)^k
\end{aligned}
$$

Multiple Testing – α Correction



- Use adjusted $\hat{\alpha}$ (Sidak Correction):

$$
\begin{aligned}
& P(\text{at least one type I error}) = \alpha \\
\longleftrightarrow \quad & 1 - (1 - \hat{\alpha})^k = \alpha \\
\longleftrightarrow \quad & \hat{\alpha} = 1 - (1 - \alpha)^{1/k}
\end{aligned}
$$

- Bonferroni Correction: use $\hat{\alpha} = \alpha/k \approx 1 - (1 - \alpha)^{1/k}$.

Dimension Reduction via Statistical Testing

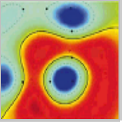
- Embedding space high-dimensional but sparse
- Some dimension do not carry real information:
  - Fixed *protocol* tokens
  - Random, *volatile* tokens (cookies, nonces, ...)
- Focus the analysis by splitting the feature set F:

$$F = F_{protocol} \cup F_{alphabet} \cup F_{volatile}$$

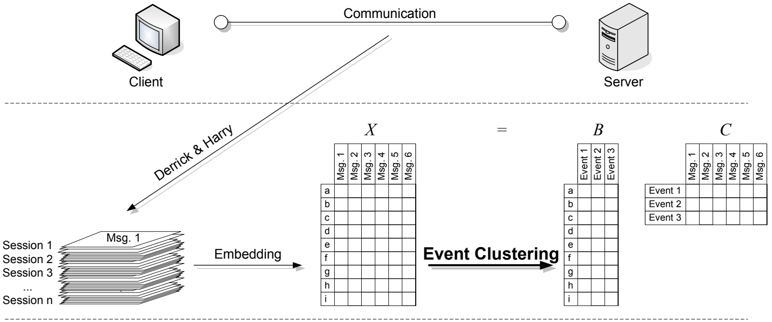
- Calculate frequency f of each feature and test via aproximated binomial test:

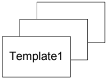
$$\begin{aligned} p_{protocol} &= binom.test(H_0 : f \approx 1.0) \\ p_{volatile} &= binom.test(H_0 : f \approx 0.0) \end{aligned}$$

- Adjust significance level α for multiple testing
- Keep features, which are not part of the protocol **and** are not volatile: $p_{protocol} \leqslant \hat{\alpha} \wedge p_{volatile} \leqslant \hat{\alpha}$

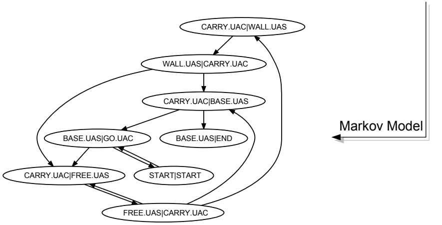


System Overview – Event Clustering

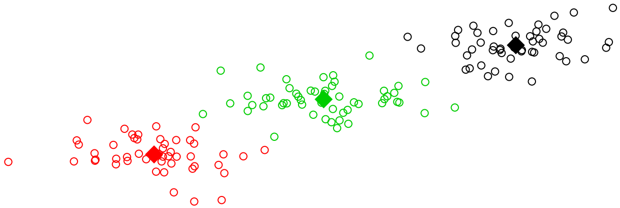




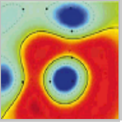

```
RULE srcField:0 dstField:0 type:ExactRule
RULE srcField:1 dstField:2 type:ExactRule
RULE srcField:3 dstField:1 type:ExactRule
RULE dstField:4 type:DataRule
data:678537408252460955, ...
RULE dstField:5 type:DataRule
data:252460957,252461242,252461608, ...
RULE dstField:6 type:DataRule
data:12004,12020,12018,12028,12002,12004
```



Clustering – Introduction



- Vectorial representation of messages allows application of geometrical concepts
- Example k-means: find k cluster *centers*, which exhibit the minimal squared *distance* to their assigned observations
- Other methods from machine learning readily applicable



Event Clustering – Application in PRISMA

- Clustering as factorization of embedding matrix $X \in \mathbb{R}^{k,N}$ with $B \in \mathbb{R}^{k,e}$, $C \in \mathbb{R}^{e,N}$, $b_i \in \mathbb{R}^{k,1}$, $c_i \in \mathbb{R}^{e,1}$, $\underline{e \ll k}$:

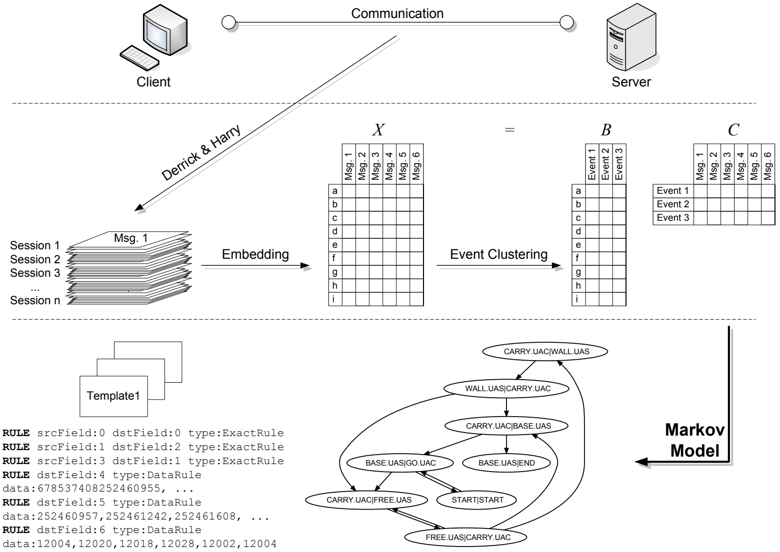
$$X \approx BC = \overbrace{\begin{bmatrix} b_1 & \dots & b_e \end{bmatrix}}^{\text{event basis}} \underbrace{\begin{bmatrix} c_1 & \dots & c_N \end{bmatrix}}_{\text{event assignments}}$$

  via *Non-Negative Matrix Factorization*:

$$\begin{aligned}(B, C) \quad &= \quad \operatorname*{arg\,min}_{B,C} \|X - BC\| \\ & \quad\quad \text{s.t. } b_{ij} \geqslant 0,\ c_{jn} \geqslant 0\,.\end{aligned}$$

- Other techniques (e.g. hierarchical clustering, expert knowledge) can be incorporated easily

System Overview – Markov Model



Markov Model – Introduction

- Probabilistic model to describe process evolving over time
- Formally, the process is a sequence of random variables:

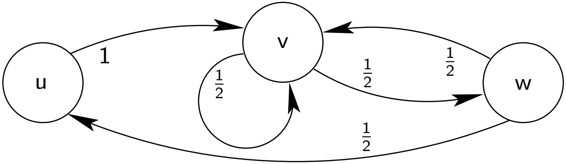
$$S_1^T = [S(1), S(2), \ldots, S(T-1), S(T)]$$

- ... which fullfill the *Markov Assumption*:

$$\forall t \in 1, \ldots, T : P_{S(t)|S(1),S(2),\ldots,S(t-2),S(t-1)} = P_{S(t)|S(t-1)}.$$

- $S(i)$ represents the internal state of the system
- Examples and notation from *Hidden Markov Models and Dynamical Systems* by Andrew M. Fraser (SIAM, 2008)

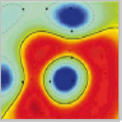
Markov Model – Example



$$P_{S(1)} = \left[\tfrac{1}{3}, \quad \tfrac{1}{3}, \quad \tfrac{1}{3}\right], \tag{1}$$

$P(s(t+1) \mid s(t))$		$S(t+1)$ u	v	w
$S(t)$	u	0	1	0
	v	0	$\frac{1}{2}$	$\frac{1}{2}$
	w	$\frac{1}{2}$	$\frac{1}{2}$	0

(2)



Markov Model – Example

- Calculate the probability of $s_1^4 = [u, v, w, v]$:

$$
\begin{aligned}
P(u, v, w, v) &= P(v|u, v, w) \cdot P(w|u, v) \cdot P(v|u) \cdot P(u) && (3) \\
&= P(v|w) \cdot P(w|v) \cdot P(v|u) \cdot P(u) && (4) \\
&= \frac{1}{2} \cdot \frac{1}{2} \cdot 1 \cdot \frac{1}{3} = \frac{1}{12}. && (5)
\end{aligned}
$$

- Eqn. (3): Conditional probability ($P_{A,B} = P_{B|A} P_A$)
- Eqn. (4): *Markov assumption*
- Eqn. (5): Eqn. (1) and Eqn. (2)
- General case for s_1^T:

$$
\begin{aligned}
P(s_1^T) &= P(s(1)) \prod_{\tau=2}^{T} P(s(\tau)|s_1^{\tau-1}) \\
&= P(s(1)) \prod_{\tau=2}^{T} P(s(\tau)|s(\tau-1))
\end{aligned}
$$

Hidden Markov Model – Introduction

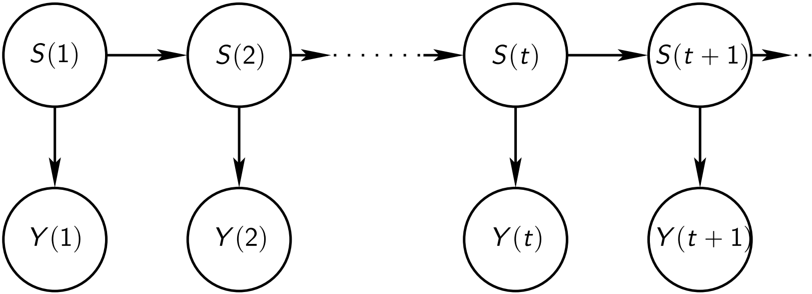
- Again, the process is a sequence of (unobservable) random variables $S_1^T = [S(1), S(2), \dots, S(T-1), S(T)]$, which generate a sequence of random variable $Y_1^T = [Y(1), Y(2), \dots, Y(T-1), Y(T)]$
- The observations are just dependent on the current *hidden* state:

$$P_{Y(t)|S_1^t, Y_1^{t-1}} = P_{Y(t)|S(t)}. \tag{6}$$

- The *hidden* state sequence is generated according to the *Markov assumption*:

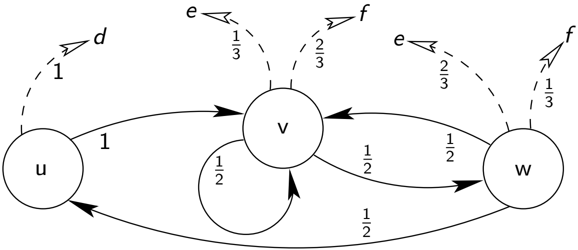
$$P_{S(t+1)|S_1^t, Y_1^t} = P_{S(t+1)|S(t)}. \tag{7}$$

Hidden Markov Model – Introduction



- Hidden Markov model as a Bayes' net
- Edges indicate dependence relations, i.e. for all $t \in 1, \ldots, T$:
  - $Y(t)$ just depends on $S(t)$
  - $S(t)$ just depends on $S(t-1)$

Hidden Markov Model – Example



		Y		
	$P(y\|s)$	d	e	f
	u	1	0	0
S	v	0	$\frac{1}{3}$	$\frac{2}{3}$
	w	0	$\frac{2}{3}$	$\frac{1}{3}$

Hidden Markov Model – Example

- Calculate the probability $y_1^4 = [d, e, f, e]$
- Possible state sequences, which could generate the observation: $[u, v, v, v]$, $[u, v, v, w]$, and $[u, v, w, v]$

s_1^4	$P(s_1^4)$	$P(y_1^4 \mid s_1^4)$	$P(y_1^4, s_1^4)$
$uvvv$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3}$	$\frac{2}{324}$
$uvvw$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{2}{3}$	$\frac{4}{324}$
$uvwv$	$\frac{1}{3} \cdot 1 \cdot \frac{1}{2} \cdot \frac{1}{2}$	$1 \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3}$	$\frac{1}{324}$

- Now:

$$P(y_1^4) = \sum_{s_1^4} P(y_1^4, s_1^4) = \sum_{s_1^4} P(y_1^4 | s_1^4) P(s_1^4) = \frac{2+4+1}{324}.$$

Hidden Markov Model – Example

Assumptions of Eqn. (6) and Eqn. (7):

$$P(s_1^T) = P(S(1)) \prod_{t=2}^{T} P(s(t)|s(t-1))$$

$$P(y_1^T|s_1^T) = \prod_{t=1}^{T} P(y(t)|s(t))$$

Single calculation:

$$\begin{aligned} P(y_1^T, s_1^T) &= P(s_1^T)\, P(y_1^T|s_1^T) \\ &= P(s(1)) \prod_{t=2}^{T} P(s(t)|s(t-1)) \prod_{t=1}^{T} P(y(t)|s(t)). \end{aligned}$$

Iterate over all possible state sequences:

$$P(y_1^T) = \sum_{s_1^T \in S^T} P(y_1^T, s_1^T)$$

Hidden Markov Model – Algorithms

- **The Viterbi Algorithm:** Given a model θ and a sequence of observations y_1^T, finds the most probable state sequence $\hat{s}_1^T$:

$$\hat{s}_1^T = \arg\max_{s_1^T} P\left(s_1^T | y_1^T, \theta\right)$$

- **The Baum-Welch Algorithm:** Given a sequence of observations y_1^T and an initial set of model parameters θ_0, calculates a new set of parameters θ_1 that has higher likelihood:

$$P\left(y_1^T | \theta_1\right) \geqslant P\left(y_1^T | \theta_0\right)$$

Hidden Markov Model – Viterbi Algorithm

- Find the *best* sequence $\hat{s}_1^T$ that maximizes the probability $P\left(s_1^T | y_1^T\right)$
- Equivalent to maximizing $\log\left(P\left(y_1^T, s_1^T\right)\right)$, since $P(y_1^T)$ is just a constant:

$$\begin{aligned}
\hat{s}_1^T &\equiv \arg\max_{s_1^T} P(s_1^T | y_1^T) \\
&= \arg\max_{s_1^T} \left(P(s_1^T | y_1^T) \cdot P(y_1^T)\right) \\
&= \arg\max_{s_1^T} \left(P(y_1^T, s_1^T)\right).
\end{aligned}$$

- Trick for numerical stability: use log

Hidden Markov Model – Viterbi Algorithm: Definitions

$$
\begin{aligned}
u(s_1^t) \quad & \text{Utility of state sequence } s_1^t \\
& \equiv \log\left(P(y_1^t, s_1^t)\right) \\
\nu(s,t) \quad & \text{Utility of best sequence ending with } s(t) = s \\
& \equiv \max_{s_1^t : s(t) = s} u(s_1^t) \\
\omega(s, s', t) \quad & \text{Utility of best sequence with } s(t-1), s(t) = s, s' \\
& \equiv \max_{s_1^t : s(t-1) = s \wedge s(t) = s'} u(s_1^t) \\
B(s', t) \quad & \text{Best predecessor state given } s(t) = s' \\
& \equiv \arg\max_{s} \omega(s, s', t)
\end{aligned}
$$

Hidden Markov Model – Viterbi Algorithm: Overview

- *Initialize* utility $\log\left(P_{Y(1),S(1)}\left(y(1), s\right)\right)$ for each state $s \in \mathcal{S}$.
- *Forward step:* for each successive time step $t : 1 < t \leqslant T$
  - for each state s determine the best predecessor for that state and store it in $B(s, t)$
  - calculate utility of the best state sequence ending in that state and store it in $\nu(s, t)$
- *Backtrack step:*
  - identify the best final state as $\hat{s}(T) = \arg\max_s \nu(s, T)$
  - for t from $T - 1$ to 1 backtrack through B array to find the other states in the sequence $\hat{s}_1^T$, i.e., $\hat{s}(t) = B(\hat{s}(t+1), t+1)$

Hidden Markov Model – Viterbi Algorithm: Iteration

Initialize:

for each s

$$\nu_{\text{next}}(s) = \log\left(P_{Y(1),S(1)}\left(y(1), s\right)\right)$$

Iterate:

for t from 2 to T

$$\#\ \nu_{\text{old}}(\cdot) = \nu(\cdot, t-1); \nu_{\text{next}}(\cdot) = \nu(\cdot, t)$$

$$\nu_{\text{old}} = \nu_{\text{next}}$$

for each s_{next}

for each s_{old}

$$\omega(s_{\text{old}}, s_{\text{next}}) = \nu_{\text{old}}(s_{\text{old}}) + \log\left(P(s_{\text{next}}|s_{\text{old}})\right) + \log\left(P(y(t)|s_{\text{next}})\right)$$

\# Find best predecessor

$$B(s_{\text{next}}, t) = \arg\max_{s_{\text{old}}} \omega(s_{\text{old}}, s_{\text{next}})$$

\# Update ν

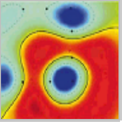
$$\nu_{\text{next}}(s_{\text{next}}) = \omega(B(s_{\text{next}}, t), s_{\text{next}})$$

Hidden Markov Model – Viterbi Algorithm: Backtrack

Backtrack:

$\bar{s} = \arg\max_s \nu_{\text{next}}(s)$
$\hat{s}(T) = \bar{s}$
for t from $T-1$ to 1
    $\bar{s} = B(\bar{s}, t+1)$
    $\hat{s}(t) = \bar{s}$

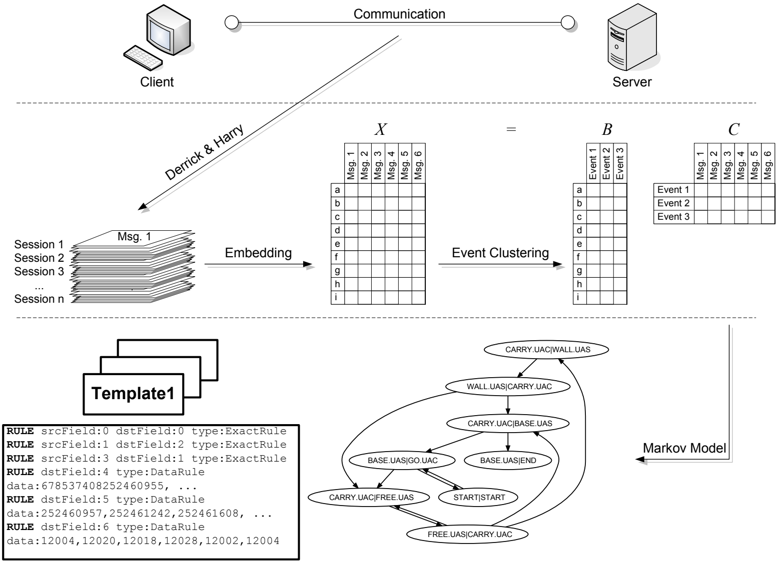
- A lot more to learn about Markov models:
  - forward/backward algorithm
  - Baum-Welch algorithm
  - Kalman filter
- So consult: *Hidden Markov Models and Dynamical Systems* by Andrew M. Fraser (SIAM, 2008)



Markov Model – Application in PRISMA

- Each message is assigned to an *event* from the event space E, so a session $S = [e_1, e_2, \ldots e_{|S|}], e_{1,2,\ldots,|S|} \in E$
- Represent the dynamics for the system by a Markov model of order $m \geqslant 2$:
  1. Estimate the frequencies of the initial events (i.e. $P(e), e \in E$)
  2. Estimate the frequencies of an event given the m predecessors in time (i.e. $P(e_t | e_{t-m}, \ldots, e_{t-2}, e_{t-1})$)
- Resulting networks can be big (potentially $|E|^m$ nodes):
  - Markov model can be transformed in a DFA
  - Compress structure via DFA minimization algorithm
- Reduced network can be described by a *hidden* Markov model

System Overview – Templates and Rules



Templates and Rules

	State A	State B	State C
Session 1	`GO 1`	`OBJECT A`	`CARRY A 0`
Session 2	`GO 2`	`OBJECT D`	`CARRY D 0`
	⋮	⋮	⋮
Session n	`GO 0`	`OBJECT B`	`CARRY B 0`
Template	`GO` $\square$	`OBJECT` $\square$	`CARRY` $\square$ `0`

- Template generation:
  - Assign each message to its corresponding state
  - Align messages and find static and changing parts (*fields*)
- Rules between templates:

Exact	copy the content of one field
Sequence	increment the number of a field
CopyCompl.	copy field and add parts before/after
CopyPartial	copy parts of the field
Data	pick a value from a data pool

Demo – Robot example

- Goal: *learn* the control of a robot, which collects goods inside a contaminated room, from network traffic
- The robot communicates with the environment by a simple protocol:
  - GO <dir>
  - CARRY <object> <dir>
- The environment responds with the following status messages after each action of the robot:
  - WALL
  - FREE
  - BASE
  - OBJECT <object>
- Hands-on example...

Demo – Robot example

Protocol Inspection and State Machine Analysis (PRISMA)

Tammo Krueger
Hugo Gascon
Konrad Rieck

Motivation

PRISMA

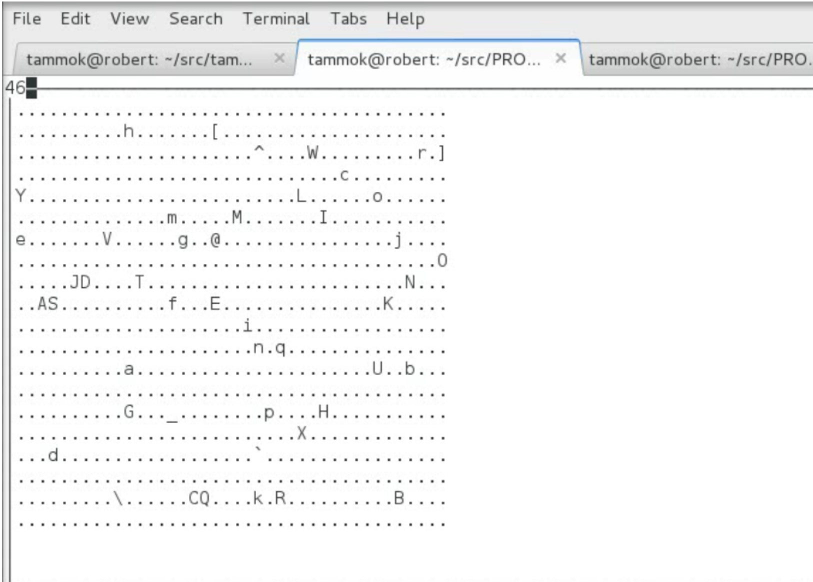
- Preprocessing
- Embedding
- Testing
- Event Clustering
- Markov Model
- Templates and Rules

Example

Evaluation

- Similarity
- Syntax & Semantic

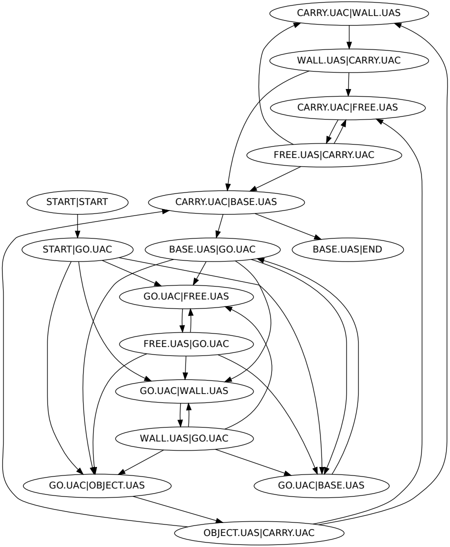
Outlook



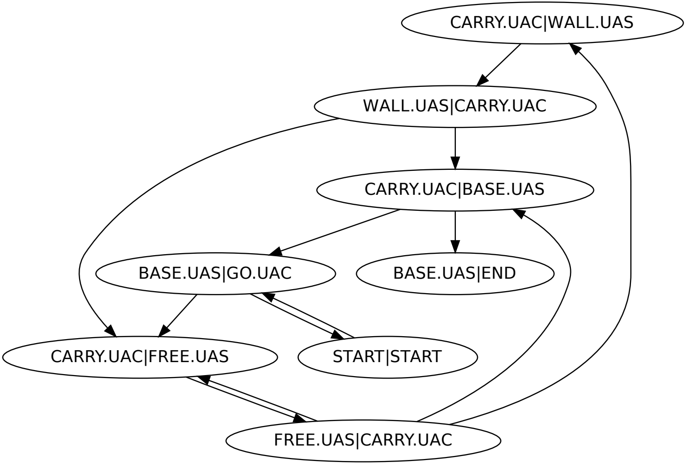
Demo – Robot example


```
GO 2 |
WALL |
GO 0 |
OBJECT E |
CARRY E 0 |
FREE |
CARRY E 0 |
WALL |
CARRY E 3 |
FREE |
CARRY E 3 |
FREE |
CARRY E 3 |
BASE |
GO 0 |
```

Demo – Robot Model before Minimization



Demo – Robot Model after Minimization



Demo – Robot Templates and Rules

TEMPLATE id:2 state:FREE.UAS—CARRY.UAC
`CARRY` □ □

TEMPLATE id:5 state:CARRY.UAC—FREE.UAS
`FREE`

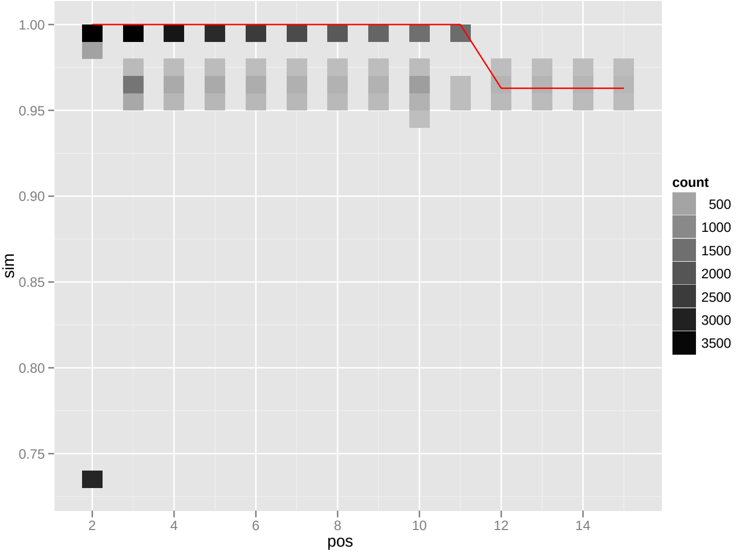
RULE transition:2;5;2
srcId:2 srcField:0 dstField:0

RULE transition:2;5;2
srcId:2 srcField:1 dstField:1

Evaluation

- Split pool into train (90% of the sessions) and testing slice
- For each session in the testing slice simulate both from the perspective of Client and Server (repeat 100 times)
- Message similarity evaluation: for each session and repetition
  1. calculate the normalized edit distance of the generated message to the real message
  2. collect all distances $\geqslant 0$ attained at a specific position
- Syntactical and semantical correctness evaluation:
  1. is message well-formed according to the underlying protocol specification (`wireshark`)
  2. is session information retained, i.e. `CallID`, `from`- and `to`-tag are preserved

Evaluation – Similarity
Alcatel-Lucent (8878 Messages)



Evaluation – Syntax & Semantic Alcatel-Lucent (8878 Messages)

	Syntax		Semantic	
	1 s. Sim.	2 s. Sim.	1 s. Sim.	2 s. Sim.
some Errors	0.03%	0.80%	3.77%	0.00%
100% Correct	99.97%	99.20%	96.23%	100.00%

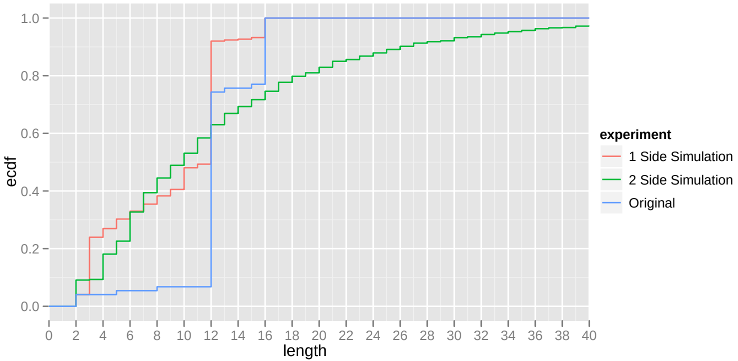
- Measure the number of correct messages per session
- **1 s. Sim.**: Simulate one side of the communication with a PRISMA model and use other side from data set
- **2 s. Sim.**: Simulate both sides of the communication with a PRISMA model

Conclusion and Future Work

- Protocol Inspection and State Machine Analysis:
  1. Embed messages in a suitable vector space
  2. Transform sequences of messages to a sequence of *events*
  3. Learn the event machine with a *Markov model*
- Application as “Honey-Service”
- Future work:
  - Stateful anomaly detection
  - Deep fuzzing
  - Infiltration of botnets

Questions? Remarks?
Thanks for your attention!

Evaluation – Length



Evaluation – Syntax & Semantic detailed Alcatel-Lucent (8878 Messages)

	Syntax		Semantic	
	1 Side Sim.	2 Side Sim.	1 Side Sim.	2 Side Sim.
$< 80\%$	0.01%	0.50%	0.30%	0.00%
$8X\%$	0.00%	0.20%	1.64%	0.00%
$9X\%$	0.02%	0.10%	1.83%	0.00%
100%	99.97%	99.20%	96.23%	100.00%

- Measure the frequency of % correct messages per session
- Reading example: For the one side simulation 0.02% of the sessions have between 90% and 99% syntactical correct messages inside the session and 99.97% of the sessions have all messages syntactical correct